Podstawy Programowania

Laboratorium 2

Standardowe algorytmy iteracyjne

prowadzący: mgr inż. Marta Lampasiak, mgr inż. Michał Jaroszczuk

1 Wprowadzenie

Celem zajęć jest poznanie podstaw programowania w języku C/C++, w szczególności ćwiczenia z tworzeniem programów ilustrujących zastosowanie instrukcji pętlowych (while, do while, for). Standardowe algorytmy iteracyjne: zliczanie, sumowanie, szukanie maksimum i minimum. Ćwiczenia z tworzeniem własnych funkcji. Funkcje bezparametrowe i zmienne lokalne. Przekazywanie parametrów przez zmienne globalne.

Podczas programowania zawsze spotykamy się z operacjami, które należy wykonywać wielokrotnie. W celu powtórzenia danego ciągu instrukcji należy skorzystać z pętli. Istnieją trzy typy pętli. Każda pętla składa się zawsze z góry określonego warunku końca oraz powtarzanych instrukcji. Poniżej pokazano w jaki sposób można wypisać ciąg liczb od 1 do 9 włącznie przy użyciu różnych pętli. Jak widać pętle można stosować zamiennie.

```
#include <iostream>
2
  int main()
3
       for (int a = 1; a < 10; a++)</pre>
5
            std::cout << a << " ";
       std::cout << "\n";
7
9
       int b = 1;
       while (b < 10)
10
           std::cout << b++ << " ";
11
       std::cout << "\n";
12
13
       int c = 1;
15
            std::cout << c++ << " ";
16
       while (c < 10);</pre>
17
       std::cout << "\n";
18
19
```

Jednak każdą pętle charakteryzuje inna cecha/zastosowanie. Pętli for używa się najczęściej w przypadku kiedy pętla ma się wykonać z góry określoną liczbę razy (dokładnie tak jak w pokazanym przypadku). W przeciwnym razie, kiedy nie wiemy ile dokładnie razy ma wykonać się pętla (np.: aż do podania konkretniej wartości), korzystamy z pętli while lub do while. Jeśli chcemy zagwarantować, aby pętla wykonała się chociaż raz to należy użyć do while, ponieważ warunek jest sprawdziany na końcu.

W celu uporządkowania oraz poprawienia czytelności programu częstym zabiegiem jest wydzielenia fragmentów programu do funkcji. W szczególności wydzielanymi fragmentami są instrukcje, które chcemy powtarzać. Należy pamiętać, że podobnie jak w przypadku definiowania zmiennych należy zawsze określić typ funkcji (void, int, bool itp.). Ponadto cechą charakterystyczną dla funkcji są nawiasy "(", ")" umieszczone bezpośrednio po nazwie funkcji. Ciało funkcji (instrukcje wewnątrz funkcji) są zgrupowane wewnątrz nawiasów "{", "}". Poprzez polecenie return można wyjść z funkcji wraz z określoną wartością. Wartość zwracana przez funkcje musi być zgodna z jej typem! W jednej funkcji polecenie return może pojawiać się więcej niż jeden raz w zależności od rozgałęzienia funkcji warunkowych.

```
#include <iostream>
2
  int a;
3
5 bool odd() {
       return a % 2;
7 }
8
   int main()
9
10 {
11
       a = 1;
12
       for (; a < 10; a++) {</pre>
            if (odd())
13
                 std::cout <<a<< " - nieparzysta\n";</pre>
15
                 std::cout <<a<< " - parzysta\n";</pre>
16
       }
17
  }
18
```

Powyżej przedstawiono kompletny kod programy, który ma zadanie określić parzystość liczb całkowitych z zakresu [1,10). Proszę zwrócić uwagę, że pole inicjalizacyjne w pętli for zostało nieuzupełnione. Pole może pozostać puste, jednak trzeba pamiętać o wszystkich średnikach.

2 Zadania

1. OCENA 3:

Napisz program z wykorzystaniem funkcji, która będzie sumować wprowadzane przez użytkownika kolejne liczby do momentu podania liczby 99. Ponadto należy sumować liczby wyłącznie z zakresu od -15 do 15 włącznie. Należy przechowywać jedynie ostatnią liczbę. Jeśli użytkownik wprowadzi następujący ciąg liczb: 20 -7 -30 2 15 3 99 to suma wczytanych liczb wynosi: 13.

2. OCENA 4:

Napisz menu użytkownika, które pozwoli na wybór programu z zadania 1 oraz obydwu programów z zadania 2. Menu powinno działać w pętli i pozwalać na wielokrotne włączanie podprogramów. Program powinien kończyć pracę poprzez wybór odpowiedniej opcji. Do wywołania poszczególnych programów wykorzystaj funkcję typu *void*.

(a) Napisz program do obliczania największego wspólnego dzielnika wprowadzonych dwóch liczb. Należy zastosować algorytm Euklidesa i przyjąć: $a, b \in C_+$

Algorithm 1: algorytm Euklidesa

```
1 function NWD(a,b):
2 | while a \neq b do
3 | if a > b then
4 | a \leftarrow a - b;
5 | else
6 | b \leftarrow b - a;
7 | return a;
```

(b) Napisz program, który wypisze "schodki" o zadanej wysokości, np. dla wysokości $n{=}5$ wygląda to następująco:

3. OCENA 5:

Napisz program tworzący prostą animację w konsoli - rysowanie zwiększającego się kwadratu. Rysowanie kształtu odbywa się poprzez stopniowe dodawanie kolejnego znaku '*' w odpowiednim miejscu, aż w końcu osiąga się cały rysunek. Następnie konsola zostaje wyczyszczona i znowu przechodzimy do rysowania kształtu, tym razem większego. Poszczególne kroki, jakie ma wykonać program zostały przedstawione na rysunku poniżej. Do wykonania zadania wykorzystaj funkcje: pozwalającą na odczekanie 1 sekundy pomiędzy rysowaniem kolejnych kształtów oraz na wyczyszczenie konsoli programu.



Rysunek 1: krok 1 Rysunek 2: krok 2 Rysunek 3: krok 3 Rysunek 4: krok 3

Podsumowując, program wykonuje poniższy algorytm:

- (a) W konsoli pojawia się rysunek 1.
- (b) Konsola zostaje wyczyszczona.
- (c) W konsoli pojawia się rysunek 2.
- (d) Konsola zostaje wyczyszczona.
- (e) W konsoli pojawia się rysunek 3.
- (f) Konsola zostaje wyczyszczona.
- (g) W konsoli pojawia się rysunek 4.
- (h) Konsola zostaje wyczyszczona.

Następnie program wraca do kroku z punktu a algorytmu.

Uwaga

Przypominam o przesłaniu programów na koniec zajęć, według wcześniej podanego wzoru. Proszę o niedołączanie plików innych niż o rozszerzeniu *.cpp.